**OS hw2**

**陈彦帆 2018K8009918002**

将实现gettimeofday的三种方法各连续执行100000遍后测量执行前后的时间，计算出一次调用的估计时间。程序如下：

#include <stdio.h>

#include <sys/time.h>

#include <unistd.h>

int main()

{

struct timeval tv1,tv2,tv3;

int time1;

gettimeofday(&tv1,NULL);

for(int i=0;i<100000;i++)

gettimeofday(&tv3,NULL);

gettimeofday(&tv2,NULL);

time1 = (tv2.tv\_sec-tv1.tv\_sec)\*1000000+(tv2.tv\_usec-tv1.tv\_usec);

printf("glibc: %.3fus\n",time1/100000.0);

gettimeofday(&tv1,NULL);

for(int i=0;i<100000;i++)

syscall(96);

gettimeofday(&tv2,NULL);

time1 = (tv2.tv\_sec-tv1.tv\_sec)\*1000000+(tv2.tv\_usec-tv1.tv\_usec);

printf("syscall: %.3fus\n",time1/100000.0);

gettimeofday(&tv1,NULL);

for(int i=0;i<100000;i++)

asm(

"mov $96, %%rax\n\t"

"syscall\n\t"

:

);

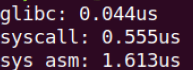
gettimeofday(&tv2,NULL);

time1 = (tv2.tv\_sec-tv1.tv\_sec)\*1000000+(tv2.tv\_usec-tv1.tv\_usec);

printf("sys\_asm: %.3fus\n",time1/100000.0);

return 0;

}

运行结果如下：

其中，glibc的函数gettimeofday速度显著快于另外两个，可能是因为gettimeofday的实现并没有用到系统调用。

尝试用更高精度的rdtsc直接计算1次系统调用的时间：代码如下：

static inline unsigned long rdtsc(void)

{

unsigned long tickl,tickh;

asm volatile (

"rdtsc\n\t"

:"=a"(tickl),"=d"(tickh));

return ((unsigned long )tickh<<32)|tickl;

}

#include <stdio.h>

#include <sys/time.h>

#include <unistd.h>

int main()

{

unsigned long tick1,tick2;

struct timeval tv1,tv2,tv3;

int time1;

tick1 = rdtsc();

gettimeofday(&tv3,NULL);

tick2 = rdtsc();

time1 = tick2-tick1;

printf("glibc: %dclocks\n",time1);

tick1 = rdtsc();

syscall(96);

tick2 = rdtsc();

time1 = tick2-tick1;

printf("syscall: %dclocks\n",time1);

tick1 = rdtsc();

asm(

"mov $96, %%rax\n\t"

"syscall\n\t"

:

);

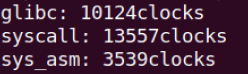
tick2 = rdtsc();

time1 = tick2-tick1;

printf("sys\_asm: %dclocks\n",time1);

return 0;

}

测试结果如下：

测试机器的基准速度为1.99GHz，即10000 clocks ~ 5us。相比多次调用的结果，直接用汇编指令syscall速度最快，可能是因为在一次调用中，前两种方式的函数调用相对开销较大。

机器信息：Linux ubuntu 5.4.0-42-generic #46~18.04.1-Ubuntu